Computer Security Symposium 2014
22 - 24 October 2014

網羅的な攻撃者モデルを考慮した Tor ブリッジ機構の強化

馮　菲　　　　松浦　幹太

東京大学生産技術研究所
153-8505　東京都目黒区駒場４－６－１
{fengfei, kanta}@iis.u-tokyo.ac.jp

あらまし Tor は世界中で広く使われている匿名通信ツールであるが、その匿名性は十分に分析されていない。公開されている中継リレーをブロックすることによるアクセス制限も Tor への脅威となりえるが、この脅威に対処するために Tor はエントリーポイントとしてブリッジと呼ばれる非公開リレーを利用することができる。しかし、現在のブリッジ機構の脆弱性はまだ十分に調査されているとは言えない。本研究ではさまざまな攻撃モデルのもとでブリッジ機構の脆弱性評価を行う。そして、現在のブリッジ機構を我々が提案する新たな二種類のブリッジ機構と比較し、Tor のセキュリティとパフォーマンスの与える影響について論ずる。

Stronger Bridge Mechanisms of Tor Considering Exhaustive Adversarial Models

Fei Feng　　　　Kanta Matsuura

Institute of Industrial Science, The University of Tokyo
4-6-1 Komaba, Meguro-ku, Tokyo, 153-8505, JAPAN
{fengfei, kanta}@iis.u-tokyo.ac.jp

Abstract Tor is the most popular anonymous communication tool in the world. Its anonymity, however, has not been thoroughly evaluated. For example, it is possible for an adversary to restrict access to the Tor network by blocking all the publicly listed relays. In response, Tor utilizes *bridges*, which are unlisted relays, as alternative entry points. However, the vulnerabilities of the current bridge mechanism have not been thoroughly investigated yet. We first investigate the vulnerabilities of the current bridge mechanism under different adversarial models. Then we compare it with two different methods and discuss their effects on security and performance of Tor.

1 Introduction

In today's expanding online world, there is an increasing concern about the protection of anonymity and privacy in electronic services. The Tor[5] network is the most popular anonymous communication system. It is a low-latency anonymity, privacy and censorship resistance network whose relays are run by volunteers around the world. Tor is used by private citizens, corporations, and governments to protect their online communications, as well as users trying to circumvent censorship. Its security is therefore essential for the safety and commercial concerns of its users.

While a common use of Tor is to protect the privacy, a growing set of Tor users use it as a tool for censorship resistance. Since the destination of a Tor's client is hard to control or determine, Tor can be effective tool for accessing to sites that some regimes may wish to block or censor. However, because the list of all Tor relays are publicly available from directory servers, blocking access to Tor is as simple as downloading the list and blocking connections to the tuples of IP/port it contains.

To counteract this situation, designers of Tor introduced a new method of accessing the Tor network: *bridges*[6], which are designed to help censored users. Bridges are Tor relays that are not listed in the main Tor directory authorities and are alternatives for publicly listed entry relays as entries into the Tor network. Since there is no complete public list of bridges, even if the ISP is filtering connections to all the known Tor relays, they probably won't be able to block all the bridges.

Effective attacks to block or attack the bridge mechanism [9, 17, 18] are also being found and conducted in the wild. However, the vulnerabilities of the current bridge mechanism have not been thoroughly investigated yet. This motivates us to construct stronger bridge mechanisms toward those attacks. We first thoroughly investigate the vulnerabilities of the current design under exhaustive adversarial models. Then we compare our proposing designs with the current design and show their effectiveness through simulation regarding to security and performance of Tor.

2 Background

2.1 Overview of Tor

The Tor network, the implementation of the second generation of the Onion Routing, aims to prevent users from being linked with their communication partners; i.e. someone monitoring a client should be unable to discover which server he/she is accessing, and the server (or someone monitoring the server) should be unable to discover the identity of the client using Tor to access it.

The Tor network is a TCP overlay network whose infrastructure is run entirely by volunteers. Tor users download and install the Tor client software, which acts as a SOCKS proxy interfacing their client software with the Tor network. The client first connects to one of the directory authorities, which monitor relays' availability and bandwidth capacity, and then periodically generates a list of status for these known Tor relays. From these authorities the client downloads the list, which is called consensus file. The client then selects three of these relays, and builds an encrypted channel, which secured by a session key established through an authenticated Diffie-Hellman key exchange, to the first relay (called the entry guard). Over this encrypted channel, the Tor client builds an encrypted channel to the middle relay, and then via this channel, connects to the third relay (called the exit relay). In this way, the client has a connection to the exit relay, but the exit relay is not aware of whom the entry guard or client is; similarly the entry guard does not know which exit relay the client has selected. Figure 1 shows the structure of the Tor network.

The client selects an exit relay, then the guard relay, finally the middle relay. There are additional constraints on the path, such as avoiding using more than one relay from a given /16 network or the same relay family. Details are available in the Tor Path Specification [16].

2.2 Bridges

Tor users can send email and instant messages, surf websites, and post content online without anyone knowing who or where they are. Consequently, it is widely acknowledged as an im-

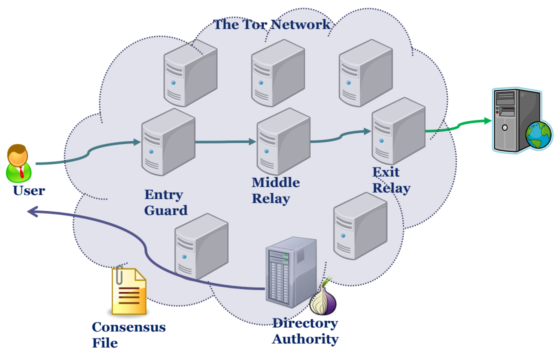


Figure 1: The structure of the Tor network.

portant tool for freedom of expression and censorship resistance. That's clearly a worry for authoritarian regimes that want to control and limit their citizens' access to the Tor network. As a list of Tor relays is publicly available from directory authorities, it is trivial for an ISP to block all connections to Tor by blocking access to IP addresses of all Tor relays[17, 18].

To counteract this situation, designers of Tor introduced a new method of accessing the Tor network: bridges[6], which are designed to help censored users. Bridges are Tor relays that are not listed in the main Tor directory authorities and are alternatives for publicly listed entry guards as entries into the Tor network. Since there is no complete public list of bridges, even if the ISP is filtering connections to all the known Tor relays, they probably won't be able to block all the bridges.

A bridge can be operated on a server or a personal computer by a Tor user who is willing to help censored users to reach the Tor network. A standard Tor client can be easily configured to operate as a bridge. Bridges can be strictly unlisted (in which case information of this bridge is spread within a group of people by word of mouth), or their descriptors can be distributed online by the Tor Project. The bridge authority keeps track of valid bridges, and the bridge database[13] distributes bridge information through the web and e-mail, which makes it easy for any client to find a few bridges. On the other hand, the distributing mechanism attempts to make it difficult for an attacker to enumerate bridges in a short time, which is realized by restricting the distribution of bridge descriptors to one set per 24-bit IP address prefix in a week.

Censored users can get several bridges by visiting the BridgeDB site[1] or send an email to bridges@bridges.torproject.org with the line "get bridges" in the body of the mail.

3 Goals and Adversarial Models

3.1 Goals

We have two major goals. As suggested above, our first goal is to investigate the current Tor network's and its bridge mechanism's vulnerabilities to several known attacks toward bridges.

Our other major goal is to propose stronger alternative bridge mechanisms. We investigate whether adopting our proposing designs helps to mitigate these known attacks. We compare the current round-robin method with the two methods we propose through simulation experiments.

3.2 Adversarial Models

Effective attacks to block or attack the bridge mechanism [9, 17, 18] are being found in existing works and also conducted in the wild. Those adversaries may have different purposes and motivations. Some of them try to enumerate bridges and block the usage of Tor, while others may want to profile or locate the bridge users. As a result, they also conduct attacks in different means - passively or actively. In order to propose stronger bridge mechanism, we first exhaustively summarize possible adversarial models. Only with these adversarial models can the vulnerabilities of the current bridge mechanism be thoroughly investigated.

Censorship. We first consider an active adversary with full control of the local network, who is capable of monitoring, injecting, replaying, shaping and dropping packets but only within his network bounds. An example is the Great Firewall as described by Wilde[17]. When a Tor user within the adversary's network bounds establishes a connection to a bridge, deep packet inspection (DPI) boxes identify the Tor protocol. Shortly after the Tor connection is detected, active scanning is initiated. The scanner pretends to be a normal Tor user and tries to establish a Tor connection to the suspected bridges. If it succeeds, the bridge will be blocked. The details of how the Great Firewall blocks are described by Winter and Lindskog[18]

Enumeration of bridges by malicious middle relays. Next, we consider a passive adversary who runs malicious Tor relays to discover bridges. This attack has been floating around in the wild, and was documented by Ling et al[9]. Normal clients use entry guards for the first node of their paths to protect them from long-term profiling attacks, but bridge users use their bridge as a replacement for the first node. As a result, if an adversary runs a relay that doesn't have the Guard flag and rejects to be an exit relay, the only position it will end up is the middle one. Then nodes that build paths to connect to this relay are normal relays and bridges. The adversary can easily identify whether the node, which connected to his malicious middle relay, is bridge or not, by referring to the public consensus files which contains all IP address of relays.

Malicious bridges. Bridge relays are donated by volunteers who are willing to help censored users. On the other hand, it is hard to trust every of them, because some of them may be operated by an attacker. In this adversarial model, we consider a passive adversary who runs malicious Tor bridges. His goal is to do traffic observing when his malicious bridges are used as the first node into the Tor network. Then the adversary may perform statistical profiling attack or fingerprinting attack[7, 11, 12] on the user. However, the concrete procedures of these attacks are out of the scope of this research. What we will investigate is the chance that these malicious bridges are used by innocent users.

Bridge set fingerprinting. It has been discussed in the Tor community that the sets of entry guards might be the fingerprint for a user[4, 8]. When a user connects to Tor from multiple locations where the network is monitored by the same adversary (e.g. a malicious network provider), his persistent use of the same set of entry guards uniquely identifies the user and shows the adversary that connections are all coming from the same user. This could also allow malicious exit nodes - in connection with other attacks - to link clients across destinations. This fingerprinting problem is also related to bridge, and it is even worse because there is guard rotation for normal Tor clients, while there is no rotation of bridges.

To investigate how vulnerable the current bridge design is under these adversarial models, and the chance that an adversary blocks or compromises Tor bridge, we conduct simulation experiments with publicly available data provided by the CollecTor[2].

4 Vulnerabilities of the Current Bridge Mechanism

4.1 Experiment Design

We have developed a Tor bridge path simulator for bridge users based on information from Tor Path Specification [16], Tor Directory Protocol [15], Tor Bridge Specification[14] and the Tor source code, which accepts the existing bridge descriptors and bridge network statuses as input. Unlike publicly available historical relay descriptors, bridge descriptors are sanitized by the Tor project by removing or replacing all potentially identifying information, because making bridge data available would defeat the purpose of making bridges hard to enumerate for censors. For example, the bridge identity is replaced with its SHA1 value. However, the sanitizing does not affect our simulation experiment because we could identify a bridge by its unique SHA value in our simulator.

Our simulator implements only Tor's relay selection logic and does not simulate the actual construction of paths, the data transmission, or network effects such as congestion. To simulate Tor's path selection precisely, the simulator generates paths with specified constraints, such as only using relays with the Exit flag for the exit position, and avoiding using more than one relay from a given /16 network. Regarding to the exit policies, we assume bridge users do web browsing which mainly utilizes port 80 and 443.

It is worth mentioning that, in our simulator, clients use the measured bandwidth in consensus files when choosing relays, while evaluates the performance that users experience with bridges' or relays' advertised bandwidth recorded in descriptors, which is the same way as what the current Tor clients do. As mentioned in the Tor source code, when weighting bridges, they enforce 20KB/s as lower and 100KB/s as upper bound of believable bandwidth, because there is no way for them to verify a bridge's bandwidth currently.

Our simulator not only simulates paths but also bridge users' clients, in all of which there are N configured bridges chosen from all the bridge network statuses of February 2014. The number N varies under different scenarios. In the following experiments, we use the bridge network status and relay consensus of the Tor network on 28th February 2014 as input, to simulate the bridge users and adversary on that day.

All the historical data used in our experiment are downloaded from CollecTor[2]. We run the simulator on a 8-core 3.20GHz Intel Core i7 machine with 23.5GB of memory on Ubuntu 12.04 with the 3.2.0 Linux kernel.

4.2 Censorship

We first conduct the simulation of censorship events to investigate how vulnerable the current bridge mechanism is to censorship events. There is an active adversary with full control of the local network within his network bounds. When a Tor user within the adversary's network bounds establishes a connection to a bridge, deep packet inspection (DPI) boxes identify the Tor protocol. Shortly after the Tor connection is detected, active scanning is initiated. The scanner pretends to be a normal Tor user and tries to establish a Tor connection to the suspected bridges. If the connection is built, the censor can be sure it is a bridge and thus block it.

We assume there are 200 Tor bridge users within the adversary's network bounds. In all the clients, there are 4 to 12 bridges configured (the number of bridges is chosen uniformly at random). The adversary could block N% of the online bridges used by the 200 users. We run all the experiments for three rounds, and the average values of the results are shown in Table 1.

For not all the bridges recorded in February's bridge network statuses are online on 28th February, some clients may have no access to the Tor network even if there is no censorship event. As shown by Table 1, there are 17.9% clients that have no online bridges. Then the adversary starts the active blocking of clients' online bridges. We assume he can block N% of all clients' online bridges. The results are shown in Table 1. After 75% are blocked, only fewer than 35% clients have access to the Tor network. Blocking 75% may seem like a difficult task, but it is possible if the adversary is of the scale of the Great Firewall.

Table 1: Simulation results of censorship events.

Percentage of Bridges Blocked	Percentage of Clients with No Online Bridge	Increment
0%	17.9%	0.0%
25%	28.3%	56.7%
50%	41.2%	147.9%
75%	65.8%	256.2%

4.3 Enumeration of Bridges

Since bridges are not publicly listed, adversaries who want to block usage of Tor would like to enumerate bridges. Next, we conduct simulations of an adversary who tries to enumerate bridges. This passive adversary runs malicious Tor relays to discover bridges. If the adversary runs a relay that doesn't have the Guard flag and rejects to be an exit relay, the only position it will end up is the middle one. Thus nodes that build paths to connect to malicious middle relays are normal relays and bridges, because normal clients use entry guards for the first node of their paths, while bridge users use their bridge as a replacement for the first node. The adversary can easily identify whether these nodes are bridges or not, by referring to the public consensus files which contains all IP address of relays.

We simulate 20,000 Tor bridge clients and 5 Tor paths for every clients. In all the clients, there are 4 to 12 bridges configured (the number of bridges is chosen uniformly at random). The adversary runs X malicious middle relays. We run all the experiments for three rounds, and the average values of the results are shown in Table 2.

On 28th February, there are 3014 bridges online. As shown by Table 2, when the adversary controls 200 malicious relays, over 80% unique bridges running on that day will be enumerated. This attack is not very costly because the adversary can rent IPs on Amazon EC2.

Table 2: Simulation results of enumeration of bridges by malicious middle relays.

Number of Malicious Middle Relays	Number of Found Bridges	Percentage of Found Bridges
50	690	22.88%
100	1085	36.01%
150	1497	49.67%
200	2414	80.08%

4.4 Malicious Bridges

It is hard to trust every bridge, because they are donated by volunteers, and thus some of them may be operated by an attacker. In this adversarial model, we consider a passive adversary who runs malicious Tor bridges. His goal is to do traffic observing when his malicious bridges are used as the first node of a Tor path. Then the adversary may perform statistical profiling attack or fingerprinting attack[7, 11, 12] on the user.

We simulate 20,000 Tor bridge clients and 5 Tor paths for every clients. In all the clients, there are 4 to 12 bridges configured (the number of bridges is chosen uniformly at random). The adversary runs Y malicious bridges. If one or more paths of the five paths is started with a malicious bridge, the client is defined as compromised. The results are shown in Table 3. This attack could also be performed by renting IPs on Amazon EC2 with comparatively low cost.

Table 3: Simulation results of clients compromised by malicious bridge.

Number of Malicious Bridges	Number of Compromised Clients	Percentage of Compromised Clients
0	0	0.00%
50	548	2.74%
100	1112	5.56%
150	1611	8.06%
200	2094	10.47%

4.5 Bridge Set Fingerprinting

It has been discussed in the Tor community that the sets of entry guards might be the fingerprint for a user[4, 8]. When a user connects to Tor from multiple locations where the network is monitored by the same adversary, his set of entry guards uniquely identifies the user and shows the adversary that connections are all coming from the same user. This fingerprinting problem is also related to bridge, and it is even worse because there is guard rotation for normal Tor clients, while there is no rotation of bridges.

We simulate 20,000 Tor bridge clients and 5 Tor paths for every clients. In all the clients, there are 4 to 12 bridges configured (the number of bridges is chosen uniformly at random). We run the experiments for four rounds, and the average number of distinct bridge sets is 12,633. Thus, the number of expected users per set is 1.58, which means every user's guard set is distinct with high probability.

It is obvious that the more bridges one knows, the less likely his client has no online bridge. On the other hand, the more bridges one knows, the more likely the bridge set is unique. In the following experiments, we assume every client is configured with the same number of bridges, to investigate the relationship between the number of bridges, the number of unique bridge set, and the number of clients that have no online bridge. We simulate 100,000 Tor bridge clients for every experiment. The results are shown in Table 4. It is clear from these results that there is trade-off between availability and the possibility of being fingerprinted. We consider 7 is the optimal number, because when 7 bridges are configured, over 80% clients have access to the Tor network and the number of expected users is 2 which ensures most clients will not have unique bridge set. However, this is not the real case because in Tor network every client knows different number of bridges. What we suggest is that, if a bridge user knows over 7 bridges, just configure 7 in the client, and periodically change the set of bridges manually.

5 Possible Countermeasures

In the previous section, we investigate the vulnerabilities of the current bridge mechanism under four different adversarial models. In this section, we propose two alternative methods of round-robin over all bridges, and investigate whether these methods make the bridge mechanism more robust toward the aforementioned attacks compared to the current round-robin method.

Table 4: Simulation results of clients with different number of bridges.

Number of Bridges	Number of Clients that have no Online Bridges	Number of Bridge Sets	Percentage of Clients that can Access the Tor Network	Number of Expected Users per Set
3	49416	14295	50.6%	7.00
4	39430	22371	60.6%	4.47
5	30936	31133	69.1%	3.21
6	24669	39453	75.3%	2.53
7	19486	47780	80.5%	2.09
8	15367	55263	84.6%	1.81
9	12229	61733	87.8%	1.62
10	9655	68103	90.3%	1.47
11	7683	73065	92.3%	1.37
12	5965	77888	94.0%	1.28

5.1 Alternative Methods

Currently, if ten bridges are configured, the client round-robins over all of them. Instead of the round-robin method, we propose two alternative methods.

- **Top one method:** The first method is to stick to the first bridge configured, and move to the next one only when the previous one is offline. The client will return to utilize the first one when it becomes online again.
- **Top three method:** The second method is to imitate the current entry guard mechanism. Instead of choosing a new guard every time, normal Tor client (non-bridge user) maintains a list, which is called guard list, of several (by default, three) pre-sele cted guards. When a path is constructed by the Tor client, the entry relay to be used is selected uniformly at random from the client's guard list. Our second proposal is to imitate this mechanism. When there are over three bridges configured, the client randomly chooses one of the first three bridges (the first/ first two brigdges, if there are fewer than three) when a path is being constructed.

5.2 Comparison

We conduct simulation experiments to compare the current round-robin method with the two alternatives under these adversarial models except the censorship model. Our proposals can not mitigate such active attack that identifies Tor protocol, but researches have been done on how to make Tor's traffic undetectable[3, 10, 19].

First, we assume there is an adversary who runs 100 malicious bridges and simulate 20,000 clients and 3 paths for every client respectively to investigate how much clients he can compromise under three different methods. Let's recall that if at least one path is started with a malicious bridge, the client is defined as compromised. Table 5 shows when the round-robin method is used, clients are most likely to be compromised. And sticking to the first one is the safest method under this adversarial model. It is easy to understand that sticking to the first one can decrease the possibility of a client exposed to malicious bridges.

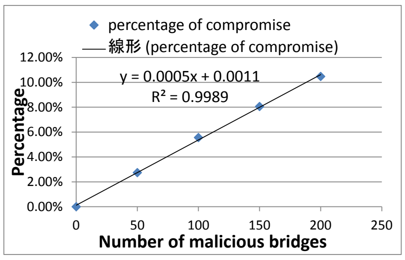
Table 5: Comparison of three methods under malicious bridge model.

	Number of Compromised Clients	Percentage of Compromised Clients
Round-robin	1019	5.10%
Top one	523	2.62%
Top three	934	4.67%

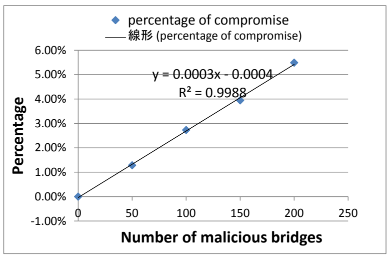
Then we change the number of malicious bridges under three methods respectively, and find the linear approximation of the relation between the number of malicious bridges and percentage of compromised clients in Figure 2. Figure 2 shows that results are almost linear, and they also show that under the adversarial model of malicious bridges, sticking to the first one bridge is the safest method, while the current round-robin method is most vulnerable one.

Next, we compare the three methods under bridge set fingerprinting model. We simulate 20,000 clients and 5 paths for every client respectively. The results in Table 6 show that sticking to the first bridge can significantly mitigate this problem, while using the top three bridges does little help compared to the original round-robin method.

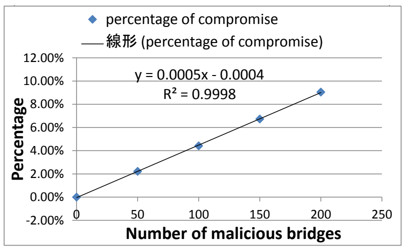
We plot the cumulative distribution functions of the size of expected anonymity set, that is the number of expected users per bridge



(a) Round-robin.



(b) Top one bridge.



(c) Top three bridges.

Figure 2: Linear approximation of the relation between the number of malicious bridges and percentage of compromised clients under three different methods.

Table 6: Comparison of three methods under bridge set fingerprinting model.

	Number of Distinct Bridge Sets	Number of Expected Users per Set	Number of Clients with a Unique Bridge Set
Round-robin	12655	1.58	10775
Top one	3004	6.66	86
Top three	12642	1.58	10765

set, for the three methods respectively, in order to analyze the probability of being fingerprinted. As Figure 3 shows, when there are 20,000 bridge users, sticking to the first one can ensure the median user an anonymity set of 6 users. On the other hand, in the round-robin or top three bridge methods, over 85% bridge sets only has one user and over 53.8% clients have their own unique bridge set. As a result, this could allow malicious exit nodes - in connection with other attacks - to link clients across destinations, and malicious network providers to link mobile clients across locations.

Then we compare the three methods under the enumeration of bridges by malicious middle relays model. We simulate 20,000 clients and 5 paths for every client for three rounds. The results in Table 7 show that there is no direct relationship between the number of bridges found by the adversary and the methods used by clients. We consider it is because now we assume that the adversary just insert N malicious non-guard non-exit relays without considering bandwidth. However, the Tor path selection favors relays with high bandwidth[16]. Controlling relays that are more possible to be chosen as middle relays perhaps can enumerate more bridges, which is a possible avenue for our future work.

Finally, we investigate whether our proposals will negatively affect the performance. When a fast bridge is selected, as long as the middle and exit relays are not slow, the client can experience fast service. As mentioned in Section 4.1, the simulator weights bridges by enforcing lower and upper bound of their bandwidth in the network status file, but evaluates the performance by their advertised bandwidth from bridge descriptor files. We use average bandwidth of all online bridges configured in a client as the metric for evaluating performance for the round-robin method, the bandwidth of the first bridge for the top one bridge method, and the average bandwidth of the first three bridges for the top three method. We simulate 20,000 clients and 5 paths for every client respectively for three rounds. Table 8 shows that the top one method does not negatively affect the performance, while top three method causes a trivial degradation in performance. The reason of this trivial degradation is still unknown, and we consider it as a future task.

Table 7: Comparison of three methods under enumeration of bridges by malicious middle relays model.

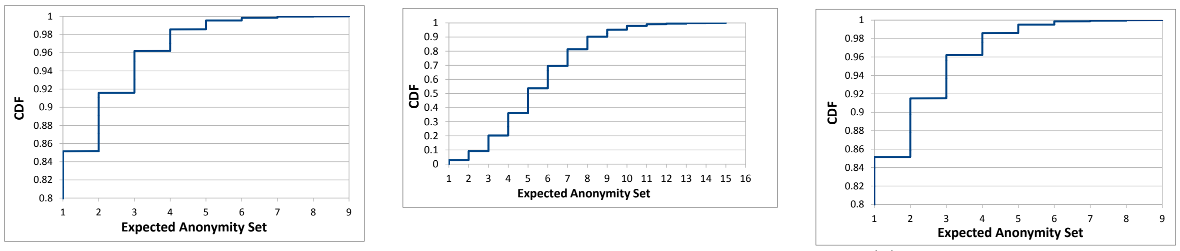
	Number of Bridges Found (Round 1)	Number of Bridges Found (Round 2)	Number of Bridges Found (Round 3)
Round-robin	826	848	577
Top one	692	965	1782
Top three	525	408	1423

Table 8: Comparison of the performance of three methods.

	Average Bandwidth (B/s) (Round 1)	Average Bandwidth (B/s) (Round 2)	Average Bandwidth (B/s) (Round 3)
Round-robin	50747	50781	50696
Top one	50251	50903	50785
Top three	49042	49109	49289

6 Conclusions and Future Work

In this research, we first investigate the vulnerabilities of the current bridge mechanism



(a) Round-robin. (b) Top one bridge. (c) Top three bridges.

Figure 3: Distribution of anonymity set size under three different methods.

under four adversarial models through simulations using our bridge path simulator. Then we compare it with our two proposals. We discover our proposal that sticking to the first bridge can effectively mitigate the attacks under malicious bridge model and bridge set fingerprinting model. And this method does not negatively affect the performance.

As mentioned before, in this research the two proposals seem not helpful in mitigating the enumeration of bridges by malicious middle relays. We regard proposing countermeasures toward this attack as our future work.

Acknowledgement

This work was partly supported by JSPS KAKENHI Grant Number 25280045.

References

[1] BridgeDB. `https://bridges.torproject.org/`

[2] CollecTor, `https://collector.torproject.org/index.html`. Accessed August 2014.

[3] Kevin P. Dyer, Scott E. Coull, Thomas Ristenpart, and Thomas Shrimpton. 2013. Protocol misidentification made easy with format-transforming encryption. In Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security (CCS '13). ACM, New York, NY, USA, 61-72.

[4] Roger Dingledine, Nicholas Hopper, George Kadianakis, Nick Mathewson. One Fast Guard for Life (or 9 months). 7th Workshop on Hot Topics in Privacy Enhancing Technologies (HotPETs 2014). Amsterdam, Netherlands, July 18, 2014.

[5] Roger Dingledine, Nick Mathewson, and Paul Syverson. 2004. Tor: the second-generation onion router. In Proceedings of the 13th conference on USENIX Security Symposium - Volume 13 (SSYM'04), Vol. 13. USENIX Association, Berkeley, CA, USA, 303-320.

[6] Roger Dingledine, Nick Mathewson. Design of a blocking-resistant anonymity system. Tor Project technical report. `https://www.torproject.org/svn/trunk/doc/design-paper/blocking.html`, Nov 2006. Accessed August 2014.

[7] Andrew Hintz. Fingerprinting websites using traffic analysis. Privacy Enhancing Technologies, Second International Workshop, PET 2002, San Francisco, CA, USA, April 14-15, 2002, Revised Papers 2003, 2482/2003:229-233, 2003.

[8] Implications of switching to a single guard node: some conclusions. `https://lists.torproject.org/pipermail/tor-dev/2014-March/006458.html`.Accessed August 2014.

[9] Zhen Ling, Xinwen Fu, Wei Yu, Junzhou Luo, Ming Yang. Extensive analysis and large-scale empirical evaluation of Tor bridge discovery. In Proceedings of the 31th IEEE international conference on computer communications (INFOCOM), Orlando, FL, USA, 25—30 March 2012, pp 2381—2389

[10] Hooman Mohajeri Moghaddam, Baiyu Li, Mohammad Derakhshani, and Ian Goldberg. 2012. SkypeMorph: protocol obfuscation for Tor bridges. In Proceedings of the 2012 ACM conference on Computer and communications security (CCS '12). ACM, New York, NY, USA, 97-108.

[11] Andriy Panchenko, Lukas Niessen, Andreas Zinnen, and Thomas Engel. Website fingerprinting in onion routing based anonymization networks. In Proceedings of the 10th annual ACM workshop on Privacy in the electronic society, WPES'11, pp. 103-114, New York, NY, USA, 2011. ACM.

[12] Yi Shi and Kanta Matsuura. 2009. Fingerprinting attack on the tor anonymity system. In Proceedings of the 11th international conference on Information and Communications Security (ICICS'09), Si-han Qing, Chris J. Mitchell, and Guilin Wang (Eds.). Springer-Verlag, Berlin,Heidelberg, 425-438.

[13] Tor bridgedb. `https://gitweb.torproject.org/bridgedb.git/tree`. Accessed August 2014.

[14] Tor Bridge Specification, `https://gitweb.torproject.org/torspec.git?a=blob_plain;hb=HEAD;f=attic/bridges-spec.txt`. Accessed August 2014

[15] Tor Directory Protocol, `https://gitweb.torproject.org/torspec.git/blob/HEAD:/dir-spec.txt`. Accessed April 2014.

[16] Tor Path Specification, `https://gitweb.torproject.org/torspec.git?a=blob_plain;hb=HEAD;f=path-spec.txt`. Accessed August 2014.

[17] Tim Wilde. Great Firewall Tor Probing Circa 09 DEC 2011. `https://gist.github.com/da3c7a9af01d74cd7de7`, Dec 2011. Accessed August 2014.

[18] Philipp Winter, Stefan Lindskog. How the Great Firewall of China is Blocking Tor. In Proceedings of 2nd USENIX Workshop on Free and Open Communications on the Internet, FOCI '12, Bellevue, WA, USA, August 6, 2012.

[19] Zachary Weinberg, Jeffrey Wang, Vinod Yegneswaran, Linda Briesemeister, Steven Cheung, Frank Wang, and Dan Boneh. 2012. StegoTorus: a camouflage proxy for the Tor anonymity system. In Proceedings of the 2012 ACM conference on Computer and communications security (CCS '12). ACM, New York, NY, USA, 109-120.